

Chapter X

Automatic Control and Behavior Of Actors

Daniel Thalmann

*Computer Graphics Lab, Swiss Federal Institute of Technology
Lausanne, Switzerland*

1. Introduction

An important part of the current animation consists in simulating the real world. To achieve a simulation, the animator has two principal techniques available. The first is to use a model that creates the desired effect. A good example is the growth of a green plant. The second is used when no model is available. In this case, the animator produces "by hand" the real world motion to be simulated. Until recently most computer-generated films have been produced using the second approach: traditional computer animation techniques like keyframe animation, spline interpolation, etc. Automatic motion control techniques (Wilhelms 1987) have been proposed, but they are strongly related to mechanics-based animation and do not take into account the behavior of characters. However, high level animation involving human beings and animals may be produced using behavioral and perception models. Reynolds (1987) introduced the term and the concept of behavioral animation in order to describe the automatization of such higher level animation.

This new approach of computer animation has less and less to do with the techniques of traditional animation but more and more with the techniques of actor direction. The animator is responsible for the design of the behavior of characters from path planning to complex emotional interactions between characters. His job is somewhat like that of a theatrical director: the character's performance is the indirect result of the director's instructions to the actor. The computer director directs at the video screen synthetic actors, decors, lights and cameras using a natural language. If it is in real time, it will be like directing a real film but in a synthetic world. We will enter into the era of real computer-generated films, produced in a virtual world and directed by real human directors. The use of synthetic actors will be interesting. Due to the personality of the actors, their reactions may sometimes cause

surprises. For example, it should be almost impossible (as in a theatrical scene) to exactly play the same scene twice. You cannot walk exactly the same way from the same bar to home twice.

A synthetic actor (Magenat Thalmann and Thalmann 1987) is defined as a human-like autonomous actor completely generated by computer. Applications of synthetic actors are unlimited: in the near future, any human being, dead or alive, may be recreated and placed in any new situation, with no dependence on any live action model. Digital scene simulation will be possible for landscapes with human beings, cinema or theater actors, and spatial vessels with humans; any human behaviour may be simulated in various situations, scientific as well as artistic. From a user point-of-view, TV announcers may be simulated, or people may walk inside their dream house before the house is built.

However, the problems to be solved to achieve this are also numerous and unlimited: computer-generated motions must be as natural as possible. Unfortunately, we know that walking, object grasping, and true personality are very complex to model. Researchers will need years to be able to represent real looking and behaving synthetic actors. And if we will not have all synthetic actors behaving in the same way, we have to introduce interactive psychological description capabilities.

Then new problems arise: how to model the personality, the intelligence? We need to know concrete and mathematical models of domains which still are very soft and not quite formally described. This may be the challenge in modelling human using computers for the coming century.

In this chapter, we discuss the various methods available for the direction of these actors. We classify these methods into three categories: local control, global control and autonomy. We illustrate this classification by several examples of systems developed in our laboratory: the TRACK system, biomechanics walking and vision-based behavioral animation. We also explain how to compose motions using the coach-trainee methodology.

2. Control versus autonomy

There are a lot of methods for controlling motion of synthetic actors. For example, Zeltzer (1985) classifies animation systems as being either guiding, animator-level or task-level systems. Magenat Thalmann and Thalmann (1991) propose a new classification of computer animation scenes involving synthetic actors both according to the method of controlling motion and according to the kinds of interactions the actors have. A motion control method specifies how an actor is animated and may be characterized according to the type of information it privileged in animating the synthetic actor. For example, in a keyframe system for an articulated body, the privileged information to be manipulated is the angle. In a forward dynamics-based system, the privileged information is a set of forces and torques; of course, in solving the dynamic equations, joint angles are also obtained in this system, but we consider these as derived information. In fact, any motion control method will eventually have to deal with geometric information (typically joint angles), but only geometric motion control methods explicitly privilege this information at the level of animation control. Plate 1 and 2 shows examples of synthetic actors.

The nature of privileged information for the motion control of actors falls into three categories: geometric, physical and behavioral, giving rise to three corresponding categories of motion control method.

- The first approach corresponds to methods heavily relied upon by the animator: rotoscoping, shape transformation, parametric keyframe animation. **Synthetic actors are locally controlled.** Methods are normally driven by geometric data. Typically the animator provides a lot of geometric data corresponding to a local definition of the motion. For example, we may consider rotoscoping, a method which uses sensors to provide coordinates of specific points of joint angles of a real human for each frame. Also keyframe systems are typical of systems that manipulate angles; from key angles at selected times, they calculate angles for intermediate frames by interpolation. Inverse kinematic methods may be also considered as being in this category. They determine values of joint angles from values of end effectors. The extension of the principle of kinematic constraints to the imposition of trajectories on specific points of the body is also of geometric nature. With the advent of Virtual Reality devices and superworkstations, brute force methods like rotoscoping-like methods tend to come back.
- The second way guarantees a realistic motion by using kinematics and dynamics. The problem with this type of animation is controlling the motion produced by simulating the physical laws which govern motion in the real world. The animator should provide physical data corresponding to the complete definition of a motion. Kinematic-based systems are generally intuitive and lack dynamic integrity. The animation does not seem to respond to basic physical facts like gravity or inertia. Only the modeling of objects as they move under the influence of forces and torques can be realistic. Forces and torques cause linear and angular accelerations. The motion is obtained by the dynamic equations of motion relating the forces, torques, constraints and the mass distribution of objects. Typical physical motion control methods for single actors which consider no other aspect of the environment animate articulated figures through forces and torques applied to limbs. The physical laws involved are mainly those of mechanics such as the Newton-Euler equations, the Lagrange equation, the Gibbs-Appel equation or the D'Alembert principle of Virtual Work. As trajectories and velocities are obtained by solving equations, we may consider **actor motions as globally controlled.** Functional methods based on biomechanics are also part of this class.
- The third type of animation is called behavioral animation and takes into account the relationship between each object and the other objects. Moreover the control of animation may be performed at a task-level, but we may also consider **the actor as an autonomous creature.** In fact, we will consider as a behavioral motion control method any method consisting in driving the behavior of this actor by providing high-level directives indicating a specific behavior without any other stimulus. A typical example is the definition of a command to impose a degree of fatigue on an actor like suggested by Lee et al. in their method of Strength Guided Motion (Lee et al. 1990).

In summary, we may design a scene involving an actor walking among obstacles using the three approaches. For example

1. using a keyframe system: by interpolating the values of parameters at given key times.
2. using a dynamic-based system: by calculating the positions from the motion equations obtained with forces and torques.
3. using a behavioral animation system: by automatic planning of the motion of an object based on information about the environment (decor and other objects).

However, it is clear that no method is convenient and complete to produce such a realistic scene. Using keyframes, not only the task is tedious, but also it is very difficult to be sure to avoid collisions, to guarantee a respect of feet support on the floor. Moreover, even an excellent animator is unable to design a correct walking balance using keyframes.

As there is no general method applicable to complex motions, only a combination of various techniques may result in a realistic motion with a relative efficiency.

Integration of different motion generators is vital for the design of complex motion where the characterization of movement can quickly change in terms of functionality, goals and expressivity. This induces a drastic change in the motion control algorithm at multiple levels: behavioral decision making, global criteria optimization and actuation of joint level controllers. By now, there is no global approach which can reconfigure itself with such flexibility.

3. Local motion definition and edition

We have designed an interactive tool for the visualization, editing and manipulation of multiple track sequences. A sequence is associated with an articulated figure and can integrate different motion generators such as walking, inverse kinematics, and keyframing within a unified framework. The TRACK system provides a large set of tools for track space manipulations and cartesian space corrections. This approach allows an incremental refinement design combining information and constraints from both the track space (usually joints) and the cartesian space. We have dedicated this system to the design and evaluation of human motions for the purpose of animation. For this reason, we also insure the real-time display of the 3D figure motion with a simultaneous scan of the 2D tracks.

3.1. The TRACK software structure

Figure 1 represents the hierarchy of modules used to construct the TRACK system. The Motion Generator includes key framing functions, inverse kinematics, and the walking function model. At the track manipulation level, there are functions to manipulate track data structure, e.g. reading, saving, selecting, editing, filtering, zooming, and multiple sequence manipulation. The user interface is the top level. It is built with the 5D Toolkit (Turner et al. 1990). For keyframing, TRACK provides functions to define and edit the key values for each joint, then the user can display the motion in real time or ask for the posture at any time by skimming the sequence.

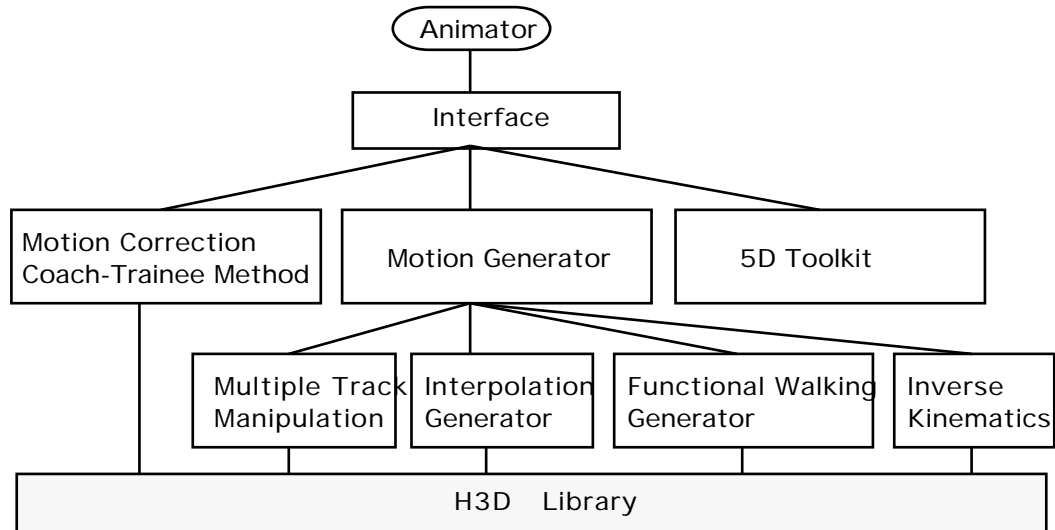


Figure 1 : the TRACK software structure

3.2 Inverse kinematics: Reaching a target with an end effector

Motion may be produced by inverse kinematics. At first, the animator picks the root and end joint, e.g. shoulder and wrist. Then, the SpaceBall or the Trackball can be used to define the target position and/or orientation interactively. The SpaceBall is a 6 DOF interactive input device. This is essentially a "force" sensitive device that relates the forces and torques applied to the ball mounted on the top of the device. These forces and torques are sent to the computer in real time where they are interpreted and may be composed into homogeneous transformation matrices that can be applied to objects. In Figure 2 example, one FREE node_3D with 6 DOF is used to represent the target, that is the frame xyz on the left of Figure 2. The position of the target can be interactively changed with the SpaceBall. First an open chain is specified with a root node_3D and an end effector node_3D. After these definitions, the end effector moves so as to reach the target by inverse kinematics. All the angles of joints between the root and the end effector are automatically evaluated.

3.3 Walking with different relative velocities

In TRACK, human walking is produced by a model built from experimental data based on a wide range of normalized velocities (Boulic et al. 1990). The user can change the current velocity in real time.

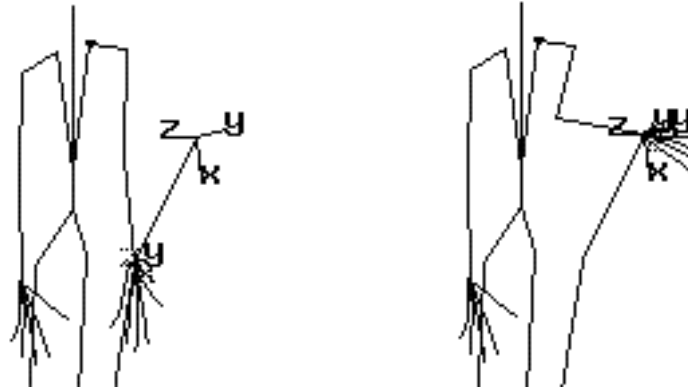


Figure 2 : Reaching the target

We have designed a new tool (Bezault et al. 1992) including path and speed profile design with real-time visualization of the step locations. This ability gives the animator both spatial and temporal control over the human figure's trajectory: for example, walking on the white squares of a chessboard or following stones in a Japanese garden path; the motions becomes easy to define. This tool is particularly efficient in that the higher level of the walking model work independently from the effective play of the low level joint trajectories by the figure. This independence is gained by means of a transfer function calibrating the figure's normalized velocity with respect to the theoretical normalized velocity. Two real-time display fonctionnalités greatly eases the design of trajectories in complex environments. First the current range of permitted velocities indicates the potentialities of the local dynamic of the walking behavior. Second the set of step locations shown on the desired path allows precise placement. Plate 3 shows an example of walking sequence.

4. Motion composition

We have introduced two types of operations: a concatenation operator and a blending operation.

4.1. Concatenation Operator

This operator concatenates a sequence A followed by a sequence B and puts the result into a sequence C. There are two parameters:

- 1) the concatenation mode manage the absolute positioning of the resulting sequence.
- A remains fixed and B is shifted after.

- B remains fixed and A is shifted before.
- A and B remain fixed.
- A is shifted so as to begin at time 0, and B is shifted after.

2) the gap value specifies the delay between A and B.

The sequences A, B and C can be the same so, combined with the mode and the gap, we provide a very powerful operator. For example it is straightforward to construct a periodical motion with 2^n periods from n successive concatenation operations.

4.2 Motion blending

Blending is necessary whenever two sequences overlap in time and we need to keep their respective time-base and migrate continuously from one to the other. We can do this by defining a normalized cubic step function which varies from 0 to 1 on the overlapping zone. This function is used to weigh the contribution of the sampled sequences on this zone for the evaluation of the blended key values. The resulting sequence is compressed afterwards to return to a standard Hermite spline form.

4.3 The Coach-Trainee Correction method

The Coach-Trainee correction method is based on a combination of direct and inverse kinematic control (Boulic and Thalmann 1992) over an open chain (no closed loop). We just recall here the basic definitions of inverse kinematic control and the principle of our motion correction approach.

Inverse kinematic control is based on a linearization of the geometric model of a chain at the current state of the system. As a consequence, its validity is limited to the neighborhood of the current state and, as such, any desired motion has to comply with the hypothesis of small movements.

The basic idea of the Coach-Trainee correction method (figure 3) is to consider the joint motion delivered by a motion generator as a reference model whose tracking is enforced through the secondary task while the main task insures the realization of desired Cartesian constraints over some specified end effector(s).

The user is more interested in local reshaping of the motion both in time and space. For this reason we prefer to use half-space constraints (planar, cylindrical, spherical). Then, we duplicate the 3D hierarchy over the controlled chain with an associated parallel control. The initial 3D hierarchy is simply controlled by the reference motion and serves as a guide for the correction control over the duplicated chain. For this reason, we refer to the reference motion as the *Coach* motion and the corrected motion as the *Trainee* motion. This *Coach-Trainee* metaphor is suggested by sport training as the adaptation of the coach reference movement into a trainee movement, being the closest possible with respect to a different context of body structure and/or Cartesian constraints.

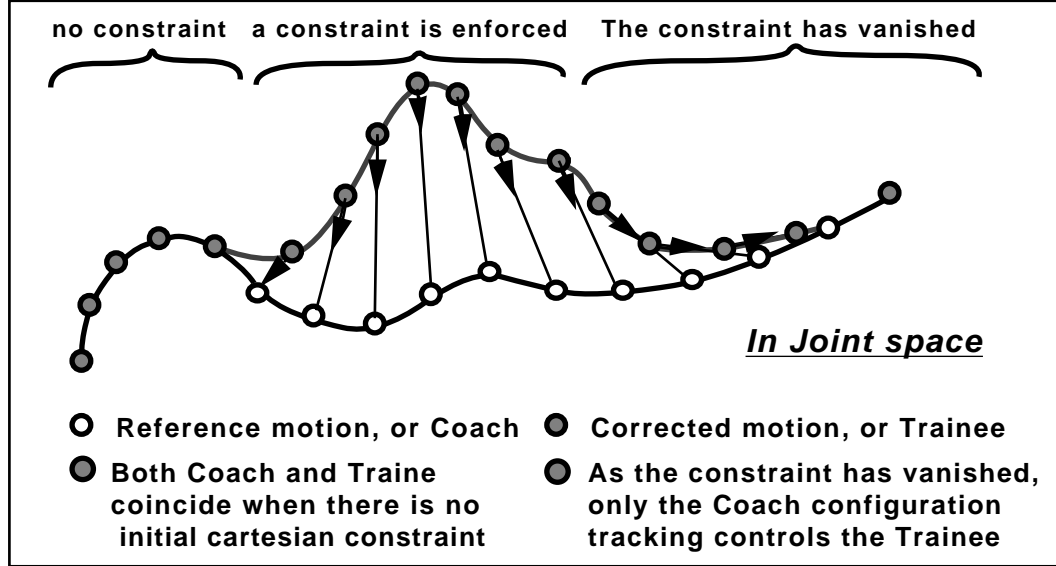


Figure 3 : The *Coach-Trainee* metaphor

Even with the Coach-Trainee parallel control, the solution presents a first order discontinuity at the interface of the half-space constraint. For this reason we confer a thickness on the boundary of the half-space constraints and we operate smooth switching on the resulting *transition zone* by means of a *transition function*. Then, the formulation of the combined "direct and inverse" kinematic control method is given by the following equation (one dimensional constraint case) :

$$= J^+ (f \mathbf{x} + (1 - f) \mathbf{J} \mathbf{z}) + (\mathbf{I} - J^+ \mathbf{J}) \mathbf{z} \quad \text{with } 0 \leq f \leq 1 \quad (1)$$

where \mathbf{z} is the unknown vector in the joint variation space, of dimension n . \mathbf{J} is the Jacobian matrix of the linear transformation, representing the differential behavior of the controlled system over the dimensions specified by the *main task*. J^+ is the unique pseudo-inverse of \mathbf{J} providing the minimum norm solution which realizes the main task. \mathbf{I} is the identity matrix of the joint variation space ($n \times n$) ($\mathbf{I} - J^+ \mathbf{J}$) is a projection operator on the null space of the linear transformation \mathbf{J} . Any element belonging to this joint variation subspace is mapped by \mathbf{J} into the null vector in the cartesian variation space. \mathbf{z} describes a *secondary task* in the joint variation space. It is usually calculated so as to minimize a cost function. f is the *transition function* (see Boulic and Thalmann 1992 for details)

5. Autonomous actors

5.1. Autonomy and behavior

Virtual humans are not only visual. They have a behavior, perception, memory and some reasoning. Behavioral human animation is a new topic consisting of developing a more general concept of autonomous actors reacting to environments and making decisions based on perception systems, memory and reasoning. Behavior is often defined as the way animals and humans act, and is usually described in natural language terms which have social, psychological or physiological significance, but which are not necessarily easily reducible to the movement of one or two muscles, joints or end effectors.

A typical human behavioral animation system, is based on the three key components:

- the locomotor system
- the perceptual system
- the organism system

A locomotor system is concerned with how to animate physical motions of virtual humans in their environment. A perceptual system is concerned with perceiving the environment. The modes of attention are: orienting, listening, touching, smelling, tasting, and looking. The organism system is concerned with rules, skills, motives, drives and memory. It may be regarded as the "brain" of the actor.

5.2 Synthetic vision

As an example of behavioral animation, we have recently introduced a way of animating actors at a high level based on the concept of synthetic vision (Renault et al. 1990). The objective is simple: to create animation involving a synthetic actor automatically moving in a room, avoiding objects and other synthetic actors. To simulate this behavior, each synthetic actor has synthetic vision for perception of the world, the sole input to his/her behavioral model.

Although the use of vision to give behavior to synthetic actors seems similar to the use of vision for intelligent mobile robots, it is quite different. This is because the vision of the synthetic actor is itself a synthetic vision. Using a synthetic vision allow us to skip all the problems of pattern recognition and distance detection, problems which still are the most difficult parts in robotics vision. However some interesting work has been done in the topic of intelligent mobile robots, especially for action-perception coordination problems. For example, Crowley (1987), working with surveillance robots states that "most low level perception and navigation tasks are algorithmic in nature; at the highest levels, decisions regarding which actions to perform are based on knowledge relevant to each situation". This remark gives us the hypothesis on which our vision-based model of behavioral animation is built.

Our approach is based on the use of Displacement Local Automata (DLA), which is similar to the concept of script introduced by Schank (1980) for natural language processing. For vision, a DLA is a black box which has the knowledge allowing the synthetic actor to move in a specific part of his environment. Examples are the DLA *displacement-in-a-corridor*, the DLA *obstacle-avoidance*, the DLA *crossing-with-another-synthetic-actor*, the DLA *passage-of-a-door* etc. This concept of DLA has the advantage of being able to increase or decrease the description grain. For example, the displacement in a corridor may be one of the uses of a more general DLA *displacement-between-two-obstructions-not-far-away-one-from-the-other*. An obstruction may be a river, a wall, a road etc. This means that the DLA may be used in a corridor, on a sidewalk or a bridge. Similarly, *to circumvent the swimming-pool* corresponds to going around an obstruction. The DLA system allows for the creation of a kind of assemblage game where, using simple and modular elements, it is possible to react to a lot of everyday simple situations. It should be noted that our DLA's are strictly algorithmic; they correspond to reflexes which are automatically performed by the adult and they only use vision as the information source.

Once we have access to DLA's, how do we use them ? How and when do we activate them and stop them? One solution consists of using a blackboard approach. The current situation is described on a blackboard and each DLA comes and sees if it has to react to the situation or not. There are two drawbacks to this approach. First, we have to add to each DLA the ability to know that it must react (this is opposed to reflex). Second, for the same situation, several DLA's may react and there is a conflict to handle. Another approach consists of using a controller which, in an unknown environment, analyzes this environment and activates the right DLA. This controller also has to handle the DLA end, the DLA errors and other messages coming from the DLA's. The controller is the thinking part of our system. It makes decision and perform the high-level actions. In any known environment, the controller should be able to activate the DLA associated with the current location. This means that an environment description has to be stored somewhere. This is the role of the navigator. From information provided by the controller, the navigator builds step by step a "map" of the environment. The navigator then gives to the controller the location of the synthetic actor in the map, and of course, it should plan the path from a known location to another known location (at a high level: e.g. go to the kitchen). Another function of the navigator is to allow the controller to change the initialization of some DLA's and anticipate the DLA changes. This description of the navigator functionalities emphasize the role of the map which should be more a logical map than a geographical map, more a discrete map than a continuous map. In fact, human being only remember discrete properties like *the tree is on the right of the car*. This suggests that to give to the behavior of the synthetic actor a maximum of believability and a maximum of developing possibilities, it is essential to give the actor a world representation similar to the representation that human beings have.

Vision simulation is the heart of our system. However, we do not attempt to recreate human vision. Our purpose is to obtain a 2-3/4 D vision, which means $2D + 1/2D + 1/4D$. The $1/2D$ is due to the fact that for each point of the 2D vision, we know the distance between the eye and the point from which it is the projection. The $1/4D$ corresponds to the fact that for each point of the vision, we know to which object it belongs (e.g. object 34, which is a table). This 2-3/4D has the advantage of avoiding all problems of pattern

recognition involved in robotic vision. Our system is implemented on IRIS 4D Workstations using an object-oriented extension of the C language. For input, we have a hierarchy containing the description of 3D objects, the environment, and the camera characterized by its eye and interest point. For output, the vision is built as a 2D array of pixels on which the actor's view of the world is projected. Plate 4 shows autonomous actors avoiding obstacles based on their synthetic vision.

Conclusion

The long-term objective of our research is the visualization of the simulation of the behavior of human beings in a given environment, interactively decided by the animator. Behavioral techniques make possible the automating of high-level control of actors such as path planning. By changing the parameters ruling this automating, it is possible to give a different personality to each actor. This behavioral approach should be a major step relatively to the conventional motion control techniques. Our main purpose is the development of a general concept of autonomous actors reacting to their environment and taking decisions based on perception systems, memory and reasoning. The animation system with autonomous actors will be based on the three key components: the locomotor system, the perceptual system, the organism system. With such an approach, we should be able to create simulations of situations such as actors moving in a complex environment they may know and recognize, or actors playing ball games based on their visual and touching perception.

Acknowledgements

The research was partly supported by "le Fonds National Suisse pour la Recherche Scientifique." The author is grateful to Arghyro Paouri and Agnes Daldegan for the design of pictures.

References

- Bezault L., Boulic R., Magnenat Thalmann N, Thalmann D (1992) An Interactive Tool for the Design of Human Free-Walking Trajectories. *Computer Animation'92*, Geneva, May 1992, Springer, Tokyo, pp.87-104.
- Boulic R, Thalmann D (1992) Combined Direct and Inverse Kinematic Control for Articulated Figures Motion Editing, *Computer Graphics Forum*, Vol. 2, No.4, October 1992, pp.189-202.
- Boulic R, Magnenat Thalmann N, Thalmann D, (1990) A Global Human Walking Model with real-time Kinematic Personification, *The Visual Computer*, Vol. 6, No 6, pp.344-358.

- Boulic R, Zhiyong H, Magnenat Thalmann N, Thalmann D (1993) A Unified Framework for the Motion Manipulation of Articulated Figures with the TRACK System (to appear in *Proc. CAD/CG 93*, Beijing)
- Crowley JL (1987) Coordination of Action and Perception in a Surveillance Robot, *IEEE Expert*, winter, pp.32-43
- Lee P, Wei S, Zhao J, Badler NI (1990) Strength Guided Motion, *Proc. SIGGRAPH '90, Computer Graphics*, Vol. 24, No 4, pp.253-262
- Magnenat-Thalmann N, Thalmann D (1987) The direction of synthetic actors in the film *Rendez-vous à Montréal*. *IEEE Computer Graphics and Applications* 7(12), pp.9-19.
- Magnenat Thalmann Nadia, Thalmann Daniel (1991) Complex Models for Visualizing Synthetic Actors. In *IEEE Computer Graphics and Applications*, Vol.11, No5, pp.32-44.
- Renault O, Magnenat-Thalmann N, Thalmann D (1990) A Vision-based Approach to Behavioral Animation, *The Journal of Visualization and Computer Animation*, Vol. 1, No1, pp.18-21.
- Reynolds C (1987) Flocks, Herds, and Schools: A Distributed Behavioral Model, *Proc.SIGGRAPH '87, Computer Graphics*, Vol.21, No4, pp.25-34
- Schank RC (1980) Language and Memory, *Cognitive Science*, Vol.4, No3, pp.243-284
- Turner R, Gobetti E, Balaguer F, Mangili A, Thalmann D, Magnenat Thalmann N (1990), An Object-Oriented Methodology using Dynamic Variables for Animation and Scientific Visualisation, *Proc. Computer Graphics International '90*, Singapore, Springer-Verlag, Tokyo, pp.317-328.
- Wilhelms J (1987) Towards Automatic Motion Control, *IEEE Computer Graphics and Applications*, Vol. 7, No 4, pp.11-22
- Zeltzer D (1985) Towards an Integrated View of 3D Computer Animation, *The Visual Computer*, Vol. 1, No4, pp.249-259

Plate 1 Synthetic actors

Plate 2 Synthetic actors

Plate 3 Walking

Plate 4 Autonomous actors avoiding obstacles based on their vision